(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 2 031 520 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
04.03.2009 Bulletin 2009/10

(51) Int Cl.:
G06F 17/30 *(2006.01)*

(21) Application number: **07017213.5**

(22) Date of filing: **03.09.2007**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IS IT LI LT LU LV MC MT NL PL PT RO SE SI SK TR
Designated Extension States:
AL BA HR MK RS

(71) Applicant: **SOFTWARE AG**
64297 Darmstadt (DE)

(72) Inventors:
- **Harbarth, Juliane**
  64347 Griesheim (DE)
- **Fiebig, Thorsten**
  68259 Mannheim (DE)
- **Winkler, Kay**
  64287 Darmstadt (DE)

(74) Representative: **Heselberger, Johannes**
Patent- und Rechtsanwälte
Bardehle . Pagenberg . Dost .
Altenburg . Geissler
Postfach 86 06 20
81633 München (DE)

Remarks:
Amended claims in accordance with Rule 137(2) EPC.

(54) Method and database system for pre-processing an XQuery

(57) The invention relates to a method of pre-processing an XQuery (10) on a XML data base comprising the steps of parsing the XQuery to obtain an abstract syntax tree (20), typing the abstract syntax tree (20) to provide at least one pointer into a schema (30; 100, 200) for XML documents of the XML data base, wherein the typing step involves the use of schema (30; 100, 200) and accumulated instance data (40; 300) of the XML data base.

Fig. 4

book	id1, id2, id3, ...
book/title	...
book/author	id1, id2, .. id16
book/author/last	id1, id2, .. id16
book/author/first	...
book/editor	id17
book/editor/last	id17
book/editor/first	...
book/editor/affiliation	...
book/publisher	...
book/price	...

300

EP 2 031 520 A1

Description

1. Technical field

[0001] The present invention relates to a method and a database system for pre-processing an XQuery.

2. The prior art

[0002] XML databases are due to their flexibility more and more important technical tools of a modem information society. The efficient retrieval of XML data in response to a query is the primary purpose of almost any database system operating on a XML data base.

[0003] Executing a query upon an XML data base is performed in multiple steps which are schematically shown in Fig. 1. At first, the raw query, which is typically defined in the XML Query Language (XQuery) is parsed, i.e. the different tokens of the query are recognized and the query is subsequently represented as a structured object, which is often referred to as Abstract Syntax Tree (AST).

[0004] In a next step the query is further processed, i.e. the AST undergoes certain changes or adornments that provide hints of how to create the query execution plan. The execution plan is a sequence of steps to be performed to obtain the query result. In this context, it is important to distinguish between compile-time and run-time steps. Only the query execution occurs at run-time, i.e. actually accesses the real data. Every other step happens at compile-time and can be considered to represent pre-processing steps. The overall purpose of any compile-time query action is to keep the actual run-time access short.

[0005] In the prior art as shown in Fig. 1, it is known to use schema data, for the typing of the query, which is part of the query pre-processing (cf. the document "XQuery 1.0 and XPath 2.0 Formal Semantics", available at http://www.w3.org/TR/xquery-semantics/). Typing tries to attach a type to every expression being part of the query. This requires an underlying type system defining which types exist. In case of queries upon an XML data base, a possible type system is provided by the W3C XML Schema. Typing a query has two main purposes. Firstly, some type errors can be detected (and better pointed at) already at compile time. The second and more important advantage is that type information provides hints for query optimization and/or execution, especially with respect to index usage. This is illustrated based on the following exemplary query:

```
for $book in input()//book
where $book/author/last ="Suciu"
return $book/title
```

[0006] This query returns all titles of books in the current collection that have an author with last name 'Suciu'. Using an index upon "author/last" the execution of the above query uses less processing time and efforts, because the index lists all documents that actually contain "Suciu" as a book's author. Looking at every document individually is therefore not needed. Only title elements of these books have to be extracted.

[0007] The query pre-processing shown in Fig. 1 finds out that an index is applicable by evaluating the schema data to make sure that the path expression that denotes the value for which the indicated condition holds, only points to a certain field. Further, the XML database must have an index defined upon that field.

[0008] However, in some situations, the use of the schema data alone for successfully pre-processing an XQuery is not sufficient, in particular if the search condition is not as simple as in the above example. As a result, in spite of the typing step in Fig. 1, substantial time and processing power will still be needed for the major parts of queries that are executed on the XML data base.

[0009] It is therefore the technical problem underlying the present invention to improve the pre-processing of an XQuery on a XML database so that the execution of a correspondingly pre-processed XQuery can be better optimized and less processing time and efforts of the hardware of the database system is eventually needed during runtime.

3. Summary of the invention

[0010] In one aspect of the invention, this problem is solved by a method of pre-processing an XQuery on a XML data base comprising the steps of parsing the XQuery to obtain an abstract syntax tree, typing the abstract syntax tree to provide at least one pointer into a schema for XML documents of the XML data base, wherein the typing step involves the use of schema and accumulated instance data of the XML data base.

[0011] According to the present invention, typing information is added during XQuery pre-processing to at least some of the expressions in the XQuery AST. The typing information is a set of pointers into element or attribute descriptions in the schema underlying the XML documents of the XML data base. Using accumulated instance data in addition to schema data allows to reduce the set of pointers to a smaller set, which in turn will reduce the number of documents to be looked at, when the query is finally executed.

[0012] In one embodiment, the accumulated instance data comprises a list of paths representing elements and/or attributes occurring in XML documents of the XML data base. Such a list facilitates the optimal use of one or more indexes on the XML documents of the XML data base.

[0013] More generally speaking, the XQuery preferably comprises an XPath expression and the last method step comprises preferably a step of identifying a set of pointers onto the schema in accordance with the XPath expression and a step of excluding pointers, which according to the accumulated instance data do not occur

in the XML data base. The XPath expression may be examined from the left to the right and may include a location step along a child or attribute axis. If so, the corresponding step of identifying a set of pointers onto the schema is preferably performed before the step of excluding pointers, which according to the accumulated instance data do not occur in the XML data base.

[0014] When the XPath expression is examined from the left to the right and includes a location step along a descendant or descendant-or-self axis, the corresponding step of identifying a set of pointers onto the schema is preferably performed after the step of excluding pointers, which according to the accumulated instance data do not occur in the XML data base.

[0015] Combining the two sources of information, namely the schema data and the instance data, in a manner, which depends on the specific location step of the XPath expression of the query, facilitates the later optimization of the query. After pre-processing and optimization, the query can be executed and the results thereof displayed to a user or stored on a storage medium.

[0016] According to a further aspect of the present invention, the second of the above indicated method steps may further comprise the calculation of the expected numbers of documents of the XML data base, which must be searched for executing the pre-processed XQuery during runtime. The expected number may be indicated as an interval between a minimum and a maximum number. Providing such a number can be valuable, since it allows to estimate the amount of time, which is - for a given hardware and software combination of a database system - needed to actually execute the query.

[0017] Additional modifications or amendments of the described method are defined in further dependent claims.

[0018] Further, the present invention concerns a computer program comprising instructions adapted to perform any of the above described methods. Such a software may be installed on any kind of hardware involved with the processing of database queries such as a mainframe, a server or client of a client-server architecture or any other kind of processing system.

[0019] Finally, the present invention relates to a database system for an XML data base comprising a search engine for XQueries, the search engine being adapted to perform any of the above described methods.

4. Short description of the drawings

[0020] In the following detailed description, presently preferred embodiments of the invention are further described with reference to the following figures:

Fig. 1: A schematic flow chart of query processing in accordance with the prior art;

Fig. 2: An exemplary schema for the XML documents of an exemplary XML data base;

Fig. 3: An additional definition for the schema of Fig. 2;

Fig. 4: An example of accumulated instance information;

Fig. 5: A schematic flow chart illustrating query processing in accordance with an embodiment of the present invention.

5. Detailed description of preferred embodiments

[0021] In the following, preferred embodiments of the invention are described. At first, a substantially simplified example for the combination of schema based information with information based on accumulated user data is presented. In a second part, the general concept and its various alternatives are generally discussed.

[0022] Fig. 2 presents an example of a XML schema 100, which defines the structure of XML documents of a XML data base. The XML schema of Fig. 2 is strictly adhering to the respective W3C recommendation, which can for example be found at http://www.w3.org/TR/xmlschema-0/. As an amendment to the schema of Fig. 2, there might be a proprietary notation to state that a certain field described in the XML schema is defined as an index. This amendment might look as shown under the reference numeral 200 in Fig. 3.

[0023] In case of a simple query as mentioned above in the introductory part, the XML schema 100 of Fig. 2 and, more specifically, the amendment 200 of Fig. 3 allow to detect during query pre-processing that the index defined in the amendment 200 can be used for efficiently executing such a query. In this case, the information provided with the path in the query is fully sufficient to point to the corresponding entry in the schema and to thereby successfully pre-process the whole query without the use of any instance data.

[0024] There are however situations where looking at the path information provided with the query and the schema does not suffice, as in the following exemplary query:

```
for $book in input()//book
where $book//last ="Suciu"
return $book/title
```

[0025] Here, the index defined in the amendment 200 upon "author/last" can not be used, since the path statement "$book//last" also points to the element "editor/last" of the XML schema.

[0026] In such a situation accumulated instance information can additionally be used for query pre-processing. For example, the information that no editor's last name is provided throughout all of the XML documents of the database would re-allow to use the index defined upon "author/last".

[0027] Generally, the term "accumulated instance information" can refer to any kind of information which el-

ements or attributes that are described in the XML schema of the XML data base actually occur in instances, or how often they occur, or even which values are realized. In the simplified example of Fig. 4, it is assumed that the accumulated instance information is a list or table 300 of paths representing all elements, which are possible in XML documents according to the schema of Fig. 2 (except for the document element that necessarily occurs in every document). Each such path gets paired with a list of document ids pointing to those documents in which that element occurs.

[0028] Returning to the above exemplary query based on a path "book//last", it might be best to design the query execution in two phases. The one document containing an editor/last element can be dealt with individually and for the rest of the data, using the index defined upon author/last is still appropriate.

[0029] The efficiency gain obtained by applying the present invention is substantial: Rather than disregarding the index defined upon "author/last" and searching through all of the XML documents in the data base, the accumulated instance data allow to retain the use of the index and to directly investigate only a single additional document of the XML data base.

[0030] Explaining embodiments of the invention now in more general terms, typing during XQuery preprocessing comprises the addition of type information preferably to each expression in the XQuery AST. The type information that is added to an expression denoting a sequence of nodes from the XML data base comprises preferably the following items:

- A set of pointers into the schema;
- For each set schema pointers, a set of documents, in which the current nodes adhering to this schema pointer are to be found;

[0031] The resulting type information is obtained by following both sources of typing information and by combining the retrieved information. Examining expressions thus leads to a navigation through both information sources. This is schematically shown in Fig. 4. Similar to Fig. 1, there is at first a parsing step for the XQuery 10 leading to the AST 20. However, in contrast to the flow chart of Fig. 1, the AST 20 is further pre-processed based on both, schema data 30 and instance data 40, which together lead to the typed query 50. Further optional optimization steps eventually provide the executable code 60 for the XQuery.

[0032] In the following, an overview is provided of possible expressions in an XQuery, and how to pre-process them based on a combination of schema and accumulated instance information. The expressions comprise among others of the following groups:

- XPath Expressions
- XPath Expressions with filter conditions
- Joins
- Sequence Operations

[0033] With respect to XPath expressions, we assume that every XPath expression starts with an expression collecting documents and continues with traversing along the following axes:

- attribute
- child
- descendant
- descendant-or-self
- parent

[0034] A location step following an axis of the XPath expression might not provide an exact name to be retrieved but also one of the following three wildcard options:

- *:name
- name:
- *

[0035] Location steps, axes and wildcards of XPath expressions are described in more detail in the W3C's XPath 1.0 Recommendation (cf. for example http://www.w3.org/TR/xpath).

[0036] Inspecting path expressions for typing purposes means examining the expression stepwise from left to right and for each location step considering what was obtained by the previous location step and which axis is used in the current location step.

[0037] With respect to an attribute or child axis, the schema is at first consulted for the next location step. Assumed the path expression up to that step could be represented by a set of schema pointers, this should hold for the expression including the current step also. The set can become bigger when wildcards are used in the next step, since one schema element pointer can lead to multiple subelement pointers, if the pattern happens to cover more than one of the element's sub-elements. The set can become smaller, if a name (or pattern) does not fit any of the element's sub-elements. After having computed the result by looking at the schema, the instance information is used to get rid of those result elements that do not occur.

[0038] When following the descendant (or descendant-or-self) axis, it is recommended to refer to the instance information first, since asking the schema to provide all possible descendants often results in very big element sets. The result must be computed back to pointers into the schema.

[0039] Following the parent axis is performed as follows: In the schema, the respective parent elements from the set of current pointers is retrieved. The set of schema pointers becomes smaller, if the previous set contained pointers to elements having the same parent.

[0040] In the instance information, each schema pointer is assigned the union of the document pointers that belonged to one of the schema pointers that was the predecessor to this schema pointer. That means that the total set of document pointers remains unchanged. For example in case of the following expression:

Input()//book/author[last="Suciu"]/..

[0041] typing the part preceding the parent step yields a schema pointer to "book/author" and a set of documents that contain the path "book/author/last". Now following the parent axis, the schema pointer is switched to the book element, but the documents that need to be scanned are still those containing book/author/last since only those documents could be found following the complete path.

[0042] Filters and where expressions are the same thing spelled differently. A filter condition can be formulated as a where clause by representing the current node by a variable: node[value = 'const'] <-> where $a/value = 'const'.

[0043] When filters are included in path expressions, it has to be inspected whether these can be used to diminish the set of schema pointers and/or to be more restrictive with respect to the number of results to be expected. If filters contain path expressions, these are evaluated. If the filter expression can be proven to be always false due to paths leading to empty results, the whole path's type is empty and typing can be dropped.

[0044] Filter can be logical expressions. If a part of an or-expression is pertaining to an empty typed expression, this part can be dropped. If a part of an and-expression is typed empty, the whole expression is dropped.

[0045] Document pointers can be used as follows: If a sub-expression in a filter yields some results (i.e. document pointers), but those pointers do not intersect with the document pointers obtained with the filtered result, the whole expression is typed empty. This is illustrated by the following example:

collection("bookshop")/bib/book[tf:containsText(title,"Web")and author]

[0046] Typing this expression with respect to schema pointers yields all results that pertain to the schema entry 'book'. Having calculated the set of all documents containing books in the previous step, the filter allows to excluding those documents which do not contain both a "book/title" path and a "book/author" path.

[0047] Typing a join means typing the return clause. As most join return clauses are generated elements which contain path expressions, typing the join means typing these path expressions. They are typed considering the additional filter criterion specified by the join criterion. This means that only those documents are retained which have an entry defined for the fields occurring in a join criterion.

for $b in collection("bib")/bib/book
for $p in collection("bookshop")/price
where $b/@isbn = $p/@isbn
return <book>{$b/title}{$p}</book>

[0048] Both the first and the second 'for' expressions are typed with respect to the schema pointers and the documents to be scanned. The expressions to be typed now are $b/title and $p. The document set is the set of book occurrences that also contain an isbn attribute, since instances of book without an isbn attribute are not taking part in the join. The $p schema pointers are the same as those for collection("bookshop")/price. The document set is the set of collection("bookshop")/price minus those documents that do not contain an isbn attribute.

[0049] Sequence operations on typed path expressions can be unions, intersections, and differences. Typing a union of two typed expressions means that the pointers into the schema are a union of the separate schema pointer sets. For each schema pointer, the set of the document pointers is the union of the document pointers that belong to this schema pointer in one or both of the subsets.

[0050] For typing an intersection, the pointers into the schema as well as the document sets are intersected. Typing a difference, finally, retains the type of the first expression.

Claims

1. A method of pre-processing an XQuery (10) on a XML data base comprising the following steps:

   a. parsing the XQuery (10) to obtain an abstract syntax tree (20);
   b. typing the abstract syntax tree (20) to provide at least one pointer into a schema (100, 200) for XML documents of the XML data base, **characterized in that**
   c. step b. involves the use of schema data (30; 100, 200) and accumulated instance data (40; 300) of the XML data base.

2. The method of claim 1, wherein step b. further provides for the at least one pointer a set of XML documents of the XML data base having at least one node adhering to the at least one pointer.

3. Method according to any of the preceding claims, wherein the accumulated instance data (300) comprises a list of paths representing elements and/or attributes occurring in XML documents of the XML data base.

4. Method of any of the preceding claims, wherein the XQuery (10) comprises an XPath expression and wherein step c. comprises a step of identifying a set of pointers onto the schema (30) in accordance with

the XPath expression and a step of excluding pointers, which do according to the accumulated instance data (40) not occur in the XML data base.

5. Method according to claim 4, wherein the XPath expression is examined from the left to the right and includes a location step along a child or attribute axis and wherein the step of identifying a set of pointers onto the schema is performed before the step of excluding pointers, which according to the accumulated instance data do not occur in the XML data base.

6. Method according to claim 3 or 4, wherein the XPath expression is examined from the left to the right and includes a location step along a descendant or descendant-or-self axis and wherein the step of identifying a set of pointers onto the schema is performed after the step of excluding pointers, which according to the accumulated instance data do not occur in the XML data base.

7. Method according to any of the preceding claims 4 - 6, wherein the XPath expression comprises a filter expression.

8. Method according to any of the preceding claims, wherein step b. further comprises the calculation of the expected numbers of documents of the XML data base, which must be searched for executing the pre-processed XQuery during runtime.

9. Method according to claim 8, wherein the expected number is indicated as an interval between a minimum and a maximum number.

10. Method according to any of the preceding claims further comprising the step of executing the pre-processed XQuery on the XML data base.

11. Computer program comprising instructions adapted to perform a method according to any of the preceding method claims 1 - 10.

12. Database system for an XML data base comprising a search engine for XQueries, the search engine being adapted to perform a method of any of the preceding method claims 1- 10.

Amended claims in accordance with Rule 137(2) EPC.

1. A method of pre-processing an XQuery (10) on a XML data base comprising the following steps:

a. parsing the XQuery (10) to obtain an abstract syntax tree (20);
b. typing the abstract syntax tree (20) to provide at least one pointer into a schema (100, 200) for XML documents of the XML data base, wherein
c. step b. involves the use of schema data (30; 100, 200) and accumulated instance data (40; 300) of the XML data base,
characterized in that:
d. the accumulated instance data (300) comprises a list of paths representing elements and/or attributes occurring in XML documents of the XML data base.

2. The method of claim 1, wherein step b. further provides for the at least one pointer a set of XML documents of the XML data base having at least one node adhering to the at least one pointer.

3. Method of any of the preceding claims, wherein the XQuery (10) comprises an XPath expression and wherein step c. comprises a step of identifying a set of pointers onto the schema (30) in accordance with the XPath expression and a step of excluding pointers, which do according to the accumulated instance data (40) not occur in the XML data base.

4. Method according to claim 3, wherein the XPath expression is examined from the left to the right and includes a location step along a child or attribute axis and wherein the step of identifying a set of pointers onto the schema is performed before the step of excluding pointers, which according to the accumulated instance data do not occur in the XML data base.

5. Method according to claim 1 or 3, wherein the XPath expression is examined from the left to the right and includes a location step along a descendant or descendant-or-self axis and wherein the step of identifying a set of pointers onto the schema is performed after the step of excluding pointers, which according to the accumulated instance data do not occur in the XML data base.

6. Method according to any of the preceding claims 3 - 5, wherein the XPath expression comprises a filter expression.

7. Method according to any of the preceding claims, wherein step b. further comprises the calculation of the expected numbers of documents of the XML data base, which must be searched for executing the pre-processed XQuery during runtime.

8. Method according to claim 7, wherein the expected number is indicated as an interval between a minimum and a maximum number.

9. Method according to any of the preceding claims further comprising the step of executing the pre-processed XQuery on the XML data base.

10. Computer program comprising instructions adapted to perform a method according to any of the preceding method claims 1 - 9.

11. Database system for an XML data base comprising a search engine for XQueries, the search engine being adapted to perform a method of any of the preceding method claims 1 - 9.

Fig. 1

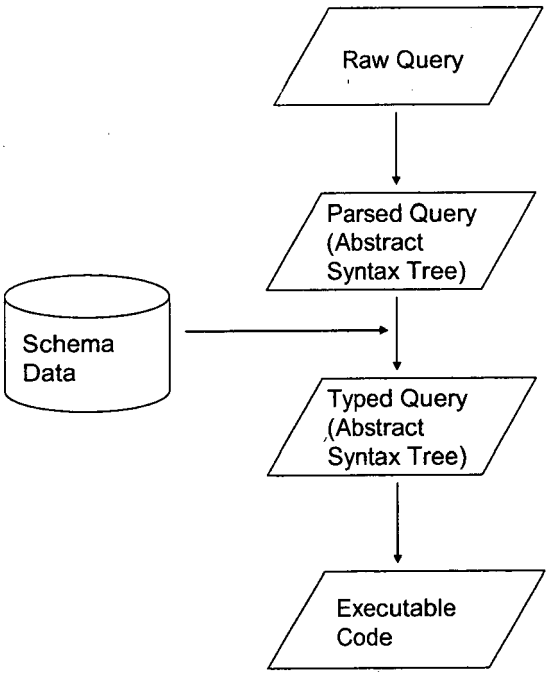

Raw Query
Parsed Query (Abstract Syntax Tree)
Schema Data
Typed Query (Abstract Syntax Tree)
Executable Code

Fig. 2

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema
      xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="bib">
 <xs:complexType>
   <xs:sequence>
    <xs:element ref="book" />
    </xs:sequence>
   </xs:complexType>
 </xs:element>

<xs:element name="book">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="title" />
    <xs:choice>
     <xs:element ref="author" maxOccurs="unbounded" />
     <xs:element ref="editor" maxOccurs="unbounded" />
      </xs:choice>
   <xs:element ref="publisher" />
   <xs:element ref="price" />
    </xs:sequence>
   </xs:complexType>
 </xs:element>

<xs:element name="author">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="last" />
   <xs:element ref="first" minOccurs="0" />
    </xs:sequence>
   </xs:complexType>
 </xs:element>

<xs:element name="editor">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="last" />
   <xs:element ref="first" minOccurs="0" /> />
   <xs:element ref="affiliation" />
   </xs:sequence>
  </xs:complexType>
 </xs:element>

<xs:element name="title" type="xs:string"/>
  <xs:element name="last" type="xs:string"/>
<xs:element name="first" type="xs:string"/>
<xs:element name="affiliation" type="xs:string"/>
<xs:element name="publisher" type="xs:string"/>
<xs:element name="price" type="xs:string"/>

</xs:schema>
```

100

Fig. 3

```
<xs:element name="author">
    <xs:complexType>
     <xs:sequence>
       <xs:element ref="last index="true">
       <xs:element ref="first" />
        </xs:sequence>
      </xs:complexType>
     </xs:element>
```

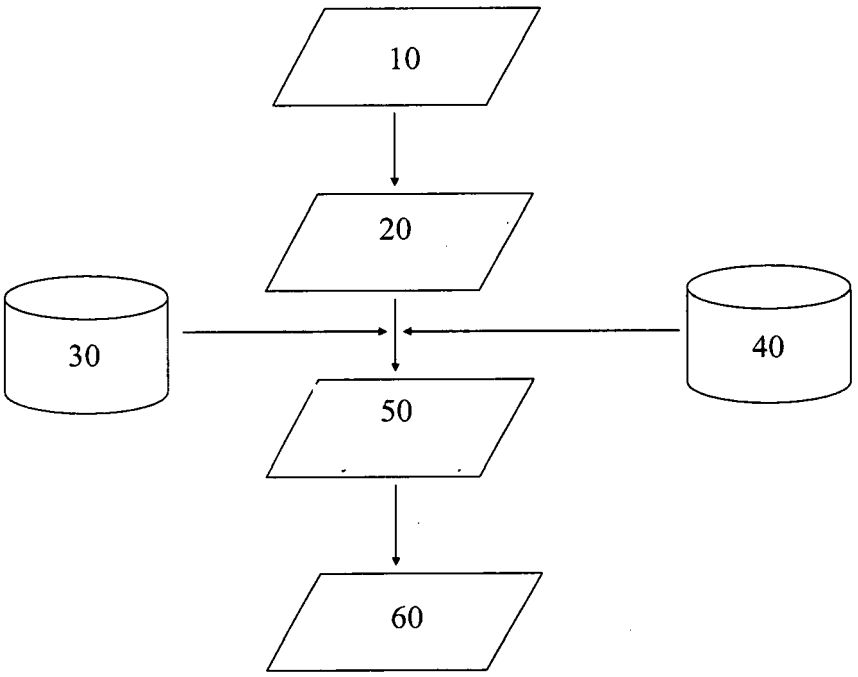
200

Fig. 4

book	id1, id2, id3, ...
book/title	...
book/author	id1, id2, .. id16
book/author/last	id1, id2, .. id16
book/author/first	...
book/editor	id17
book/editor/last	id17
book/editor/first	...
book/editor/affiliation	...
book/publisher	...
book/price	...

300

Fig. 5



European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 07 01 7213

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
E	US 2007/250527 A1 (MURTHY RAVI [US] ET AL) 25 October 2007 (2007-10-25) * paragraph [0010] - paragraph [0037] * -----	1,11,12	INV. G06F17/30
X	ZHIMAO GUO ET AL: "Index selection for efficient XML path expression processing" CONCEPTUAL MODELING FOR NOVEL APPLICATION DOMAINS. ER 2003 WORKSHOPS ECOMO, IWCMQ, AOIS, AND XSDM. PROCEEDINGS (LECTURE NOTES IN COMPUT. SCI. VOL.2814) SPRINGER-VERLAG BERLIN, GERMANY, 2003, pages 261-272, XP002459941 ISBN: 3-540-20257-9 * the whole document * -----	1-12	
X	MENG XIAOFENG ET AL: "OrientX: an integrated, schema based native XML database system" WUHAN UNIVERSITY JOURNAL OF NATURAL SCIENCES EDITORIAL DEPT. WUHAN UNIV. J. CHINA, vol. 11, no. 5, September 2006 (2006-09), pages 1192-1196, XP002459942 ISSN: 1007-1202 * the whole document * -----	1-12	TECHNICAL FIELDS SEARCHED (IPC) G06F
X	US 7 054 854 B1 (HATTORI MASAKAZU [JP] ET AL) 30 May 2006 (2006-05-30) * column 4, line 19 - column 15, line 41 * -----	1-12	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
The Hague	26 November 2007	Bernardi, Luca

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 07 01 7213

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

26-11-2007

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2007250527 A1	25-10-2007	NONE	
US 7054854 B1	30-05-2006	JP 3754253 B2	08-03-2006
		JP 2001147933 A	29-05-2001

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82